Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET)
9-10 December 2021, Cape Town-South Africa

Lossless Compression Scheme for Efficient GNSS Data Transmission on IoT Devices

Rafael Perez
Technology Department
Polytechnic Institute of Portalegre
Portalegre, Portugal
18701@ipportalegre.pt

Valderi R. Q. Leithardt
Valoriza, Polytechnic Institute of Portalegre
Portalegre, Portugal
COPELABS, ULHT
Lisbon, Portugal
valderi@ipportalegre.pt

Sérgio D. Correia
Valoriza, Polytechnic Institute of Portalegre
Portalegre, Portugal
COPELABS, ULHT
Lisbon, Portugal
scorreia@ipportalegre.pt

2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) | 978-1-6654-4231-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECET52533.2021.9698642

***Abstract*—Wireless data transmission is one of the most energy-consuming tasks performed on embedded devices, being a crucial feature of battery-powered Internet of Things (IoT) applications. The present work proposes a new methodology to reduce the energy footprint of GNSS-based sensors by decreasing the amount of transmitted data applying a lossless compression strategy. Online trajectory data is structured through a pre-processing stage without information loss and posteriorly compressed through standard lossless algorithms. Simulations are performed considering different trajectory shapes, comparing the proposed schema with traditional compression methods without the proposed pre-processing stage. The results show that the proposed scheme can reach lower compression rates, reducing embedded IoT devices' energy footprint.**

***Index Terms*—Data Transmission, Global Navigation Satellite System, Internet of Things, Lossless Compression, Wireless Sensors Networks.**

I. Introduction

Embedded systems have received increasing interest over the years, largely associated with the enormous development and penetration of the Internet of Things (IoT) in daily tasks [1]. These are also considered a core element of larger platforms such as wireless sensor networks [2], home automation [3], localization [4], [5], intelligent agricultural infrastructures [6], [7] context-sensitive systems, or solutions involving data sources related to energy management [8]–[10]. In these type of systems, in addition to the intrinsic data, and related to each specific application, the geographic location of the data source is usually considered a fundamental feature [11]. Also, a large set of these devices such as sensors and actuators are battery-operated, and thereby, power depletion becomes an imperative factor and needs an energy management approach to extend the battery life. Each sub-task such as sensing, processing, and data transmission consumes energy, which accumulates to the total power consumption [12]. Nevertheless, previous studies have demonstrated that a significant cut of the overall energy consumption of battery-powered devices is caused by network data transmission [13]–[16].

The use of data compression techniques as long be considered for bandwidth optimization [17], and more recently, for energy saving. Nonetheless, a write balance between the need of complex mathematical calculation for high performance algorithms, and the consequent obtained gain, is a complex task [18]. Data compression minimizes the size of the data packet to be stored and communicated, by reducing the size of some data file, either by using lossless or lossy compression methods. Depending on the end application, while sensory data may be considered with some losses, localization sources are less susceptible to loss of information, being more adequate to lossless compression techniques. However, compression methods usually perform unsatisfactorily in terms of space-time complexity or compression rate when considered on embedded sensors [19]. This leads to a rapid depletion of memory, storage, computing and energy resources.

Based on the above discussion and in response to the problems raised, the present work describes and demonstrates the effectiveness of a new methodology for data packaging and subsequent compression and transmission. Globally, the compression schema proposed consists of a first pre-processing stage that encapsulates the data in a JavaScript Object Notation (JSON) format, also structured to eliminate Global Navigation Satellite System (GNSS) coordinate repetitions. Essentially, while being staked in the embedded memory, a counter is incremented if some part of the coordinate system (degrees, minutes, or seconds) is held constant for both latitude and longitude. Secondly, a standard lossless compression scheme is applied before the data is transmitted. Simulations based on several different path shapes demonstrate that the proposed method archives enhanced compression rates compared with traditional deployments.

The contributions of the developed method and its implementation can be enumerated as follows: (1) it is proposed a straightforward pre-processing schema for GNSS data storage (2) and compression based on lossless algorithms, (3) where its combination shows an improved compression rate when compared with standard methodologies.

978-1-6654-1262-9/21/$31.00 ©2021 IEEE

The rest of this paper continues as follows. Section II briefly overviews existing methodologies. Section III presents the detailed methodology and discusses the selection of underlying compression algorithm. Section IV lists a set of experiments to verify the effectiveness of our method and compares it with traditional compression algorithms. Section V concludes the paper and proposes future works.

II. Related Work

Data compression consists of applying some algorithm to reduce the memory footprint of a data set on a computer system. Huffman coding [20] is one of the most popular methods for its simplicity and execution speed when considering the compression of structured data. Still, it produces poor results for models with a high degree of dispersion [21]. While in the Huffman algorithm, a probabilistic binary tree is assigned to the data set to obtain a probability of a character or sequence, other methods consider character counting. When compressed, values and positions are assigned to them. Examples of these types of algorithms are the variations LZ77 [22], and LZW [23] from the Lempel-Ziv compression model, which assigns values and indexes to characters respectively to their order of entry and repetition intervals. Also commonly used is the Run-Lenght Encode (RLE) method [24], which consists of grouping sequences of the same uninterrupted characters into a variable that represents their reference value and number of repetitions. All the mentioned methods are well established in the scientific literature are integrate several standards concerning image, video, and audio compression such as Graphics Interchange Format (GIF), Portable Network Graphics (PNG) or ZIP compression format. When compressing some data set reporting to a GNSS model, a lossless compression scheme should be applied since the sensitivity in the accuracy of the information is a value-adding component of the data set. In the case of a route, the slightest loss or data exchange may result in a deviation of several kilometers from the original path. In the present work, it is considered that the data acquisition already removes similarities and redundant coordinates. As such, complex algorithms that perform trajectory prediction or interpolation are disregarded [25]. Although complex and efficient algorithms exist to address the problem of trajectory compression based on compressed sensing [26], [27], or artificial intelligence [27], those are far too complex to integrate on embedded low power architectures, mostly with 8-bit processors.

III. Proposed Method

Two independent steps can characterize the proposed methodology. Firstly, we consider a pre-processing stage that aims to avoid many repetitions in the data packet when considering the standard data packet of degrees, minutes, and seconds of GNSS. This way, it is possible to separate the coordinate components in independent arrays and the specific analysis of each element, immediately creating savings in memory allocation, even before the compression step. Initially, the algorithm receives raw NMEA inputs, following the standard model of receiving GNSS information from sensors for embedded systems [1]. On each string entry, data is parsed to store the information in specific data vectors. Analyzing the latitude and longitude data, it is possible to keep the information of each component and work the data accordingly, having an array for each part of the coordinate point: position, degree, minute, and seconds. In each iteration cycle, the degree and minute data are also analyzed: through a logical check, where a counter has its value increased if there is a different value from the one recorded previously (Algorithm 1).

Algorithm 1 NMEA String Parser

```
1:  function PMESAGE(string *NMEA, vect *data)
2:      GLOBAL : counter, ns
3:      data[ns].ts ← NMEA.ts
4:      data[ns].pos ← NMEA.pos
5:      data[ns].deg ← NMEA.deg
6:      data[ns].min ← NMEA.min
7:      data[ns].sec ← NMEA.sec
8:      if data[ns − 1].pos! = NMEA.pos then
9:          counter.pos + +
10:     end if
11:     if data[ns − 1].deg! = NMEA.deg then
12:         counter.deg + +
13:     end if
14:     if (data[ns − 1].min! = NMEA.min then
15:         counter.min + +
16:     end if
17:     ns++                                  ▷ number of samples
18: end function
```

It is worth mentioning that, in addition to the received coordinates, the GNSS protocol also considers it possible to obtain data such as velocity and relative altitude [28]; however, for this analysis, such data were disregarded.

Once the data has been separated and analyzed, it is necessary to verify the possibility of applying encapsulation to the position, degree, and minute data depending on each counter value. The check is structured to confirm that the counters recorded more than one different value for each set. No variation in the position counter means that the displacement has not crossed any meridian line, and the encapsulation can proceed. If the degree counter has been increased twice, there are at least two different values for the set, not justifying encapsulation. Likewise, the same process applies to the minute counter as follows in the 2 Algorithm.

For the dataset implementation, the JSON format is chosen. It consists of a lightweight data standard that allows quick and easy access to information, both by embedded system and the server that receives and post-processes it. The document structure has a time variable (TS, timestamp), which defines the time of the first received coordinate and a counter for the number of coordinates that the packet brings (NS, number of samples). Also, a reference variable for how many coordinate

[1] https://www.nmea.org/

Algorithm 2 Encapsulation Analysis

function ANALYZEFX(∗**counter**)
  $GLOBALfx$
  if $counter.pos = 0$ **then**
    if $counter.deg = 0$ **then**
      if ($counter.min = 0$ **then**
        $fx = 2$ ▷ Degrees and minutes encapsulated
      else
        $fx = 1$ ▷ Degrees encapsulated
      end if
    else
      $fx = 0$ ▷ No encapsulation
    end if
  end if
end function

```
{
    "TS": int,          //Timestamp
    "NS": short int,    //Num. Samples
    "FX": byte,         //Fixed Fields
    "lat": {            //Latitude
        "PL":[byte],    //Signal
        "DL":[byte],    //Degrees
        "ML":[byte],    //Minutes
        "SL":[int fixed point]
        },              //Milliseconds
    "lng": {            //Longitude
        "PN":[byte],    //Signal
        "DN":[byte],    //Degrees
        "MN":[byte],    //Minutes
        "SN":[int fixed point]
        }               //Milliseconds
}
```

Fig. 1. JSON Structure

fields were pre-processed (named as FX) and two objects to store the latitude and the longitude data, containing in each object a matrix for geographic positioning information (maned as PL and PN), degrees (named as DL and DN), minutes (named as ML and MN) and seconds (named as SL and SN), as seen below in Figure 1.

After analyzing which fields have a repetition of values from the algorithm, it is possible to structure the JSON document to be as compact as possible, storing only information sensitive to changes along the way. Finally, verifying the encapsulation feasibility and confirming the formatting of the information set makes it possible to structure the JSON document storing only the information essential for data interpretation.

From this point, an online pre-processing stage was performed, efficiently storing all the location information and considering data portability and interruptibility through an optimized JSON scheme. At this stage, two candidate algorithms will be considered due to their simplicity and efficiency, that is Huffman Coding [20] and the LZ77 Coding [22] methods. The final step will comprise implementing some low complexity compression algorithm, where the input consists of the JSON schema developed. It should be pointed out that both methods are antagonistic because they have the duality of operation in which the Huffman model uses the probability of occurrence of a character. At the same time, the LZ77 is based on the repetition count of characters and similar sets. Therefore, avoiding repetitions using encapsulation should affect the optimal functioning of the LZ77, reducing its file compression capacity, operating with a lower number of character repetitions, but would not compromise the Huffman method since the individual probability is affected not the actual likelihood of the frequency event.

IV. Results and Discussion

Several simulation conditions were defined to assess the effectiveness, reliability, and compression rates of implementing the proposed architecture, namely incorporating the preprocessing schema on the global workflow of a standard compression algorithm. With this in mind, simulation models that can portray real displacements and practical procedures for implementing the proposed encapsulation method are considered. It is also essential that the models have similarities so that it is possible to observe the analyzed methods functioning in different displacements that follow the same pattern. It is also important that the paths represent different types of trajectories to validate the practical applicability of the proposed method.

For implementation and analysis of the results, we considered three types of paths used to simulate the data:

- Displacement in a straight path (A-B);
- ”Zigzag” displacement, covering a specific area (C-D);
- Perimeter displacement, bypassing a specific area (E-F);

Based on these three displacement patterns, two different groups of data sets were generated and simulated. One group consists of routes with few coordinates and simple displacements, while the other has a large number of coordinates and longer trajectories. There is an improvement in the quality of the simulation carried out since these two groups guarantee an excellent coverage of the types of practical displacements possible to be collected. The first group consists of 35 coordinate points per route to simulate urban scenarios with few geographic landmarks, as shown in Figure 2. This dataset aims to affect possible daily ways, implementations in familiar elements such as the rental of an electric scooter, the vigil of a neighborhood by a police vehicle, or the marking of an athlete who runs around a perimeter such as the form of training. The trajectories were first drawn from Google Earth[2] and downloaded in the Keyhole Markup Language (KML) format. A Python[3] script was created to convert the KML format to NMEA strings, with the purpose of recreating the

[2]https://www.google.com/intl/pt-PT/earth/
[3]https://www.python.org/

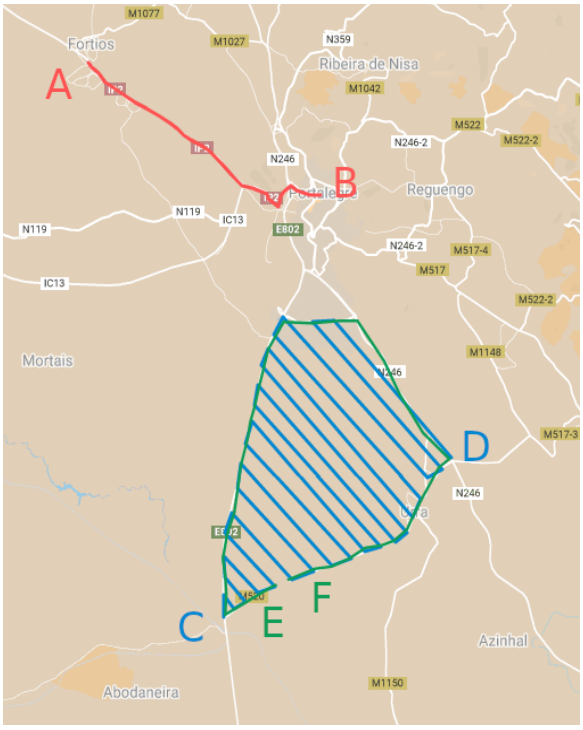

Fig. 2. Short displacements - 35 coordinate points

exact conditions of an embedded processor connected to a GNSS receiver. Algorithms 1 and 2, and the code for the Huffman and LZ77 coding algorithms were implemented in C programming language, on CodeBlocks[4] IDE, once again, with the goal of a future implementation on an embedded processor device. All simutations run on an Intel® Core™ I7-4700HQ CPU, at 2.4GHz, with 16GB of RAM, on Windows® 8 (64 Bits) operating system.

Regarding the implementation of the compression methods, Huffman and LZ77 coding, as mentioned in Section II, while the first one does not have adjustment parameters, the same does not happen with the second one. This method uses window divided to a "search buffer" and a "look-ahead" buffer, so LZ77 is sometimes called sliding-window compression. The LZ77 coding method has two parameters to be tuned. To specify the values of these parameters, an offline grid search optimization method was employed [29], [30], considering 15 different trajectories paths, with different typologies (straight path, perimeter, and "zigzag") on a $[0 \times 1000]$ cube, with a step of 10. Secondly, the grill has been refined, considering a $[0 \times 100]$ cube, with a grade of 1. Finally, the mean value of the 15 optimal test parameters was considered, and a value of 16 for the "look-ahead" buffer and 12 for the "search buffer." These values will be kept constant among all performance tests realized in this Section. Figure 3 illustrates the obtained compression rate for a straight path of $15km$ with 35 coordinate points. From Fig. 3 one can see what was a generalized behaviour of the compression rate among the different simulated conditions. While the search buffer represents a portion of the recently encoded sequence and the look-ahead pointer contains the next portion of the sequence to be encoded. Although reducing the search size, or the sliding windows it represents, reduces the compression rate, a point where no character match is found can be reached for extreme values. Regarding the look-ahead, its behaviour was not monotonous and multiple local minimums could be observed. Also, slight variations were observed between the various tests, where absolute minimum would correspond to local minimum and vice-versa. To overcome this situation, the mean value of the all test scenarios was considered has an optimal value for the look-ahead buffer.

[4]https://www.codeblocks.org/

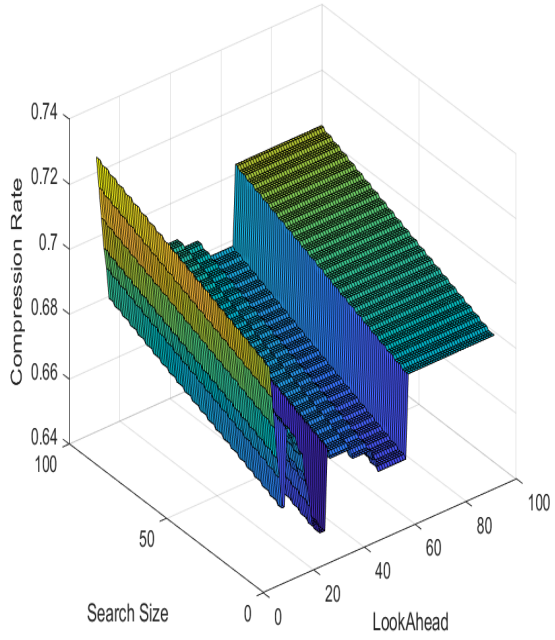

Fig. 3. Grid Search Results for a Straight Path of $15km$ with 35 Coordinate Points

TABLE I.
REDUCTION ON THE MEMORY FOOTPRINT - 35 COORDINATE POINTS

	Huffman Coding		LZ77 Coding	
	Proposed Method	Standard	**Proposed Method**	Standard
Trajectory AB	**52%**	64%	**53%**	60%
Trajectory EF	**51%**	64%	**53%**	60%
Trajectory CD	**52%**	65%	**54%**	61%
Mean	**52%**	64%	**53%**	60%
Standard Deviation	**0,0029**	0,0027	**0,0053**	0,0048

When analyzing the performance of compression algorithms, the most common metric applied is the compression rate. This is a measurement of the relative reduction in the size of data representation expressed as the quotient of uncompressed size by compressed size. Nevertheless, in our work, the inverse of the compression ratio is considered,

representing the amount of reduction in the memory footprint. Table I shows the reduction obtained when applying both Huffman and LZ77 algorithms directly to the raw data or applying the proposed methodology. Some duality can be seen between the Huffman method, which uses the probability of appearance of a character, and the LZ77 model, which is based on counting the characters present in the data. Before applying the pre-processing step, namely in the standard column, it is possible to verify a prevalence of the LZ77 model because many values are repeated, with the LZ77 having an average data reduction of 60% against 64% of Huffman (see Table I). However, after applying our encapsulation scheme, it is possible to notice that, with fewer repetitions of characters, the Huffman probabilistic model presents a marginal improvement concerning LZ77: 52% above 53%. It is also relevant to highlight the low standard deviation presented by the models, which provides better reliability as it reduces the probability of a non-standard value producing a compression result outside the desired range.

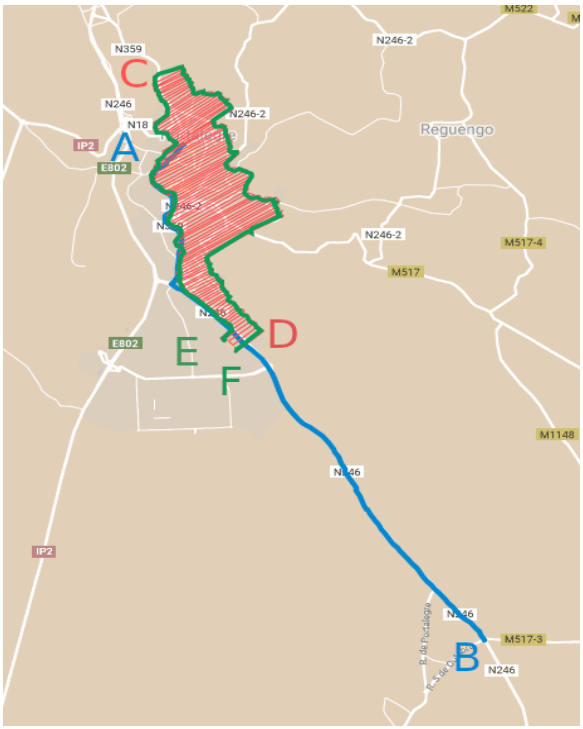

Fig. 4. Long displacements - 200 coordinate points

The second group of results deals with a path of 200 coordinate points, simulating greater displacements and wider area coverage. Real applications for this group could be aerial monitoring of forest fires by drones, tracking a hospital vehicle to a dangerous rescue, or perimeter patrolling of a major event. Figure 4 represents the considered paths. As in the case of 35 coordinate points, the same countenance is considered, where the trajectory AB represents a straight path (marked as a blue line), trajectory CD represents a ”zigzag” path (marked as a red line), and finally, trajectory CD around a certain perimeter (marked as a green line).

TABLE II.
REDUCTION ON THE MEMORY FOOTPRINT - 200 COORDINATE POINTS

	Huffman Coding		LZ77 Coding	
	Proposed Method	Standard	**Proposed Method**	Standard
Trajectory AB	**28%**	42%	**24%**	32%
Trajectory CD	**31%**	44%	**36%**	43%
Trajectory EF	**31%**	46%	**36%**	43%
Mean	**30%**	44%	**32%**	39%
Standard Deviation	**0,0141**	0,0131	**0,0557**	0,0519

Considering the results provided in Table II, it is again possible to understand the duality between the two models presented, where the LZ77 method shows a better reduction rate than the original data. Still, when considering our proposed schema, the Huffman model presents better results: a reduction of 30% against 32% of the LZ77 model, obtaining a 14% improvement in the decrease for the Huffman model, while the LZ77 reduces only 7% of the compression (as seen in Table II).

As seen in both cases, the Huffman method presents better overall performance than LZ77 after applying our encapsulation strategy. It is understood that the algorithm that uses the character frequency model stands out from the one that uses the count and sequence of characters. The encapsulation proposed in this work avoids these repetitions but does not change the total probability of the frequency event, just changing the individual probabilities of each set of characters.

V. CONCLUSION

In this work, an encapsulation strategy was developed to reduce the memory footprint of a GNSS dataset to obtain the compression of a JSON document. The search for a higher compression rate is due to the reduction in the transmission energy cost, thus optimizing the energy capacity of services that use it. The implementation of this work was given by generating a set of NMEA inputs simulating a traveled path. The algorithm then receives these inputs line by line, separating the geographic coordinate components and storing them into data vectors. Knowing that the possible repetitions would be in the data referring to cardinal points, coordinate degrees, and coordinate minutes, counters that act in each iteration cycle of the NMEA input analysis were implemented. These counters work to record the number of different entries received in this analysis set. Once all NMEA inputs have been analyzed, counters are used to define the packages to be implemented. The first verification takes place with the cardinal points to assess whether the applied displacement crosses any meridian, sensitizing the quality of the information. Once this is done, the repetitions in the data of degrees and minutes are analyzed, and, if possible, the encapsulation is performed. With the JSON document ready, compression is applied to minimize the memory space used and lower the energy cost of sending. In the process of compressing the JSON data document, it was possible to observe, after the implementation of the Huffman method and LZ77 models, the duality that exists between

the compression methods that use the global probability of occurrence of characters and the strategies that use the count of repetition of a character or sequence. The LZ77 proved to be more efficient with the Standard data model because there was a considerable repetition of characters. Still, after applying the proposed method, the Huffman coding demonstrated that its use of the probability of occurrence of a character was better suited to form a more restrictive data model. After applying the compression methods, it was possible to notice compression gain in both methods, with a marked improvement for the Huffman model, considering the duality mentioned above. When considering the encapsulation, the memory space used by the compressed set would occupy an average of 30% of the original area, a significant improvement compared to the memory space used previously, on average above 40%. As future work, implementing a test platform to analyze the practical effects of this model, such as aerial monitoring of a specific area by drone, is under consideration.

ACKNOWLEDGMENT

This work was partially supported by national funds through the Fundação para a Ciência e a Tecnologia, under projects UIDB/05064/2020 and UIDB/04111/2020; and also ILIND—Instituto Lusófono de Investigação e Desenvolvimento, under projects COFAC/ILIND/COPELABS/1/2020 and COFAC/ILIND/COPELABS/3/2020.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the internet of things: A survey," in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, 2011, pp. 1–6.

[3] W. A. Jabbar, T. K. Kian, R. M. Ramli, S. N. Zubir, N. S. M. Zamrizaman, M. Balfaqih, V. Shepelev, and S. Alharbi, "Design and fabrication of smart home with internet of things enabled automation system," *IEEE Access*, vol. 7, pp. 144 059–144 074, 2019.

[4] S. D. Correia, J. Fé, S. Tomic, and M. Beko, "Development of a testbench for evaluating the embedded implementation of the improved elephant herding optimization algorithm applied to energy-based acoustic localization," *Computers*, vol. 9, no. 4, p. 87, 2020.

[5] S. D. Correia, M. Beko, L. A. Da Silva Cruz, and S. Tomic, "Implementation and validation of elephant herding optimization algorithm for acoustic localization," in *2018 26th Telecommunications Forum (TELFOR)*, 2018, pp. 1–4.

[6] M. Waleed, T.-W. Um, T. Kamal, A. Khan, and A. Iqbal, "Determining the Precise Work Area of Agriculture Machinery Using Internet of Things and Artificial Intelligence," *Applied Sciences*, vol. 10, no. 10, 2020.

[7] V. Leithardt, D. Santos, L. Silva, F. Viel, C. Zeferino, and J. Silva, "A solution for dynamic management of user profiles in iot environments," *IEEE Latin America Transactions*, vol. 18, no. 07, pp. 1193–1199, 2020.

[8] E. M. Pinheiro and S. D. Correia, "Software model for a low-cost, iot oriented energy monitoring platform," *International Journal of Computer Science and Engineering*, vol. 5, no. 7, p. 1–5, 2018.

[9] E. M. Pinheiro and S. D. C. Correia, "Hardware architecture of a low-cost scalable energy monitor system," *International Journal of Engineering Trends and Technology*, vol. 61, no. 1, p. 1–5, 2018.

[10] E. M. Pinheiro and S. D. Correia, "Analog input expansion board based on i2c communication with plug-and-play feature, applied to current measurements," *International Journal of Electronics and Communication Engineering*, vol. 5, no. 9, p. 1–5, 2018.

[11] B. Wang, Q. Xu, C. Chen, F. Zhang, and K. J. R. Liu, "The promise of radio analytics: A future paradigm of wireless positioning, tracking, and sensing," *IEEE Signal Processing Magazine*, vol. 35, no. 3, pp. 59–80, 2018.

[12] A. Elhaddad, H. Bruckmeyer, M. Hertlein, and G. Fischer, "Energy consumption evaluation of cellular narrowband internet of things (nb-iot) modules," *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020.

[13] R. K. Singh, P. P. Puluckul, R. Berkvens, and M. Weyn, "Energy consumption analysis of lpwan technologies and lifetime estimation for iot application," *Sensors*, vol. 20, no. 17, 2020.

[14] Nasaruddin, M. Andriani, Melinda, and M. Irhamsyah, "Analysis of energy efficiency for wi-fi 802.11b multi-hop networks," in *2013 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2013, pp. 64–68.

[15] L. Guegan and A.-C. Orgerie, "Estimating the end-to-end energy consumption of low-bandwidth iot applications for wifi devices," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 287–294.

[16] B. Bougard, F. Catthoor, D. Daly, A. Chandrakasan, and W. Dehaene, "Energy efficiency of the ieee 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives," in *Design, Automation and Test in Europe*, 2005, pp. 196–201 Vol. 1.

[17] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, "Performance and power optimization through data compression in network-on-chip architectures," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, 2008, pp. 215–225.

[18] A. Gopinath and M. Ravisankar, "Comparison of lossless data compression techniques," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 628–633.

[19] L. Benini, D. Bruni, A. Macii, and E. Macii, "Hardware-assisted data compression for energy minimization in systems with embedded processors," in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, 2002, pp. 449–453.

[20] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[21] M. Nelson, *Data compression book*. Bpb Publications, 2008.

[22] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[23] Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[24] D. Salomon, *Data compression the complete reference*. Springer, 1998.

[25] P. Sun, S. Xia, G. Yuan, and D. Li, "An overview of moving object trajectory compression algorithms," *Mathematical Problems in Engineering*, vol. 2016, p. 1–13, 2016.

[26] W. Xue, C. Luo, Y. Shen, R. Rana, G. Lan, S. Jha, A. Seneviratne, and W. Hu, "Towards a compressive-sensing-based lightweight encryption scheme for the internet of things," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 3049–3065, 2021.

[27] Y. Zhu, Y. Liu, J. J. Q. Yu, and X. Yuan, "Semi-supervised federated learning for travel mode identification from gps trajectories," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[28] J. Muckell, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, "Algorithms for compressing gps trajectory data: An empirical evaluation," in *GIS '10: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Association for Computing Machinery, 2010, p. 402–405.

[29] S. Gorlatch, P. Fragopoulou, and T. Priol, *Grid computing: achievements and prospects*. Springer, 2008.

[30] S. D. Correia, M. Beko, S. Tomic, and L. A. Da Silva Cruz, "Energy-based acoustic localization by improved elephant herding optimization," *IEEE Access*, vol. 8, pp. 28 548–28 559, 2020.